Store, Carry and Forward Networks: Defining the Problem, Requirements and Expectations

William D. Ivancic *Member, IEEE,* Wesley Eddy, Joseph Ishac, Alan Hylton and Dennis Iannicca

***Abstract*—This document provides a problem statement for Store, Carry and Forward (SCF) network, a network consisting of non-realtime communication between systems that are generally disconnected, which require multiple network hops between source and destination, and which may never be fully connected end-to-end at any given time. Included as part of this problem statement are a number of use cases that motivate having a standard method to communicate between such systems, as multi-organization and multi-vendor support and interoperability is highly desirable.**

This document also describes the requirements for a SCF protocol, and the expectations placed upon the SCF agents and SCF applications as well as guidelines and requirements for testing SCF systems and protocols.

***Index Terms*—Network Architectures, Protocols, Store and Forward Networks**

CONTENTS

W. Ivancic, J. Ishac, D. Iannicca and A. Hylton are with NASA Glenn Research Center, Cleveland, Ohio

W. Eddy is with MTI Systems Inc.

I. TERMINOLOGY

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, ”SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in Request For Comment (RFC) 2119 [1].

“What’s in a Word. Words make a difference. They affect how we think about something. The terms chosen to describe a concept are a crucial part of any model. The right concepts with terms that give the wrong connotation can make a problem much more difficult. The right terms can make it much easier. Adopting the mindset of the terms may allow you to see things you might not otherwise see.” - John Day [2]

In developing this document, we have intentionally avoided some terminology used by other protocols - particularly other store-and-forward protocols - to avoid biases and confusion that may otherwise ensue.

Container - The application/user data to be transported over the network as well as a checksum of that information. Containers may include sub-containers, or be sub-containers themselves.

Container Aggregation - The process of organizing one or multiple containers as sub-containers inside another larger container.

Container Deaggregation - The process of removing one or more sub-containers from a larger container. This differs from fragmentation because rather than creating new containers, deaggregation operates on existing sub-containers.

Container Fragmentation - The process of dividing a single container's contents into multiple new containers which will need to be eventually reassembled back into the original container before delivery to the application.

Container Reassembly - The process of recombining the contents of multiple containers into the single container that they were originally part of, and which needs to be delivered to the application intact.

Delay - Propagation delay between SCF agents. Delay does not include disconnection time.

Disruption - A relatively short period of disconnection within an otherwise well-connected network, e.g. a loss of connectivity in the range of seconds to perhaps minutes.

Disconnection - A relatively long period where communication between a significant proportion of hosts is not possible for various reasons, e.g. due to the inability to close a radio link.

Metadata - Synonymous with a Container's Shipping Label

SCF - Store, Carry and Forward

SF - Store-and-Forward, or "store and forward" as used generically in other literature (where the presence of hyphenation varies)

SCF Agent - A protocol instance providing SCF services to an upper-layer user/application

Shipping Label - Metadata describing the characteristics of a container and its forwarding requirements

Sub-Container - A smaller container residing inside a larger container.

Transport Capacity - As a first order approximation, the product of link capacity and contact time.

II. Introduction and Background

INTERNET technology has become pervasive and is now present in many types of devices that end up being deployed in the field for use in scenarios where they do not have good (or any) actual Internet connectivity. The networking stacks are used to support data transfer during episodes of connectivity, and the applications and protocol configurations avoid reliance on many typical infrastructure services (e.g. Domain Name System (DNS)). For instance, these devices may be only intermittently connected to other devices, and are used to support data flows where the source and ultimate destination might never be fully connected to one another at any time. These applications operate highly asynchronously, with non-realtime constraints on their communication. Often, there are intermediate relaying nodes (or "agents") that must "carry" the data while waiting for connectivity to develop. The means for relaying data has been highly specialized in such systems (almost per-deployment), and varies widely, with little code-reuse or commonality in the supporting network design.

The"problem statement" portion of this document describes several of these scenarios generically, motivates the development of a common solution, and describes shortcomings in existing technologies. The problem statement is explicitly not trying to look at the situation where a smart phone or mobile computer is temporarily off or removed from the Internet, and then is reattached directly to the edge of a well-connected network. Such systems are well-suited to utilize standard Internet protocols and are able to support realtime communications when connected. The systems and applications that this document is concerned with are primarily operating with a much higher level of asynchrony between the data producers, individual relays, and eventual data consumers. We call these "Store, Carry, and Forward" (SCF) systems to distinguish them from typical Store and Forward (SF) systems, which generally operate over a better-connected infrastructure. This section clarifies the distinction between SCF and the better-understood SF concept, which is already implemented by a number of different networking technologies.

This document also describes the requirements for a SCF protocol, and the expectations placed upon the SCF agents and SCF applications as well as guidelines and requirements for testing SCF systems and protocols.

Because so many analogies exist between SF/SCF-based computer communications and offline non-computer-based systems, it is useful to understand (very) early communications. Relay systems have been present even in ancient civilizations, where rulers utilized intelligence gathering via courier services to convey and obtain information. Relays consisted of runners, messengers, and even pigeons conveying messages and documents. These types of relay agents had only intermittent connectivity with one another and needed to hold onto messages for possibly long amounts of time before delivering them. Later relay systems included the ancient Greeks using fires, mirrors, or colored flags for visually communicating over large distances, and the the 18th century French using a system of telescopes and semaphores. These involved relatively well-connected systems of relay infrastructure, compared to the earlier methods that involved physical carriage of the stored messages for some time in order to reach the next forwarding point. Telegraph and later systems had equally well-connected infrastructures.

In the present day, the postal system is a well-understood example of a physical store and forward system. This is well-connected system of storage and forwarding agents (mail boxes, post offices, mail carriers, trucks, airplanes, etc.) with various capabilities (storage capacity and delivery capability) and location-dependent addressing and routing.

A. Store and Forward Systems

In computer networking, numerous technologies that support SF message communications between systems have evolved, and some have incorporated pieces of what SCF systems require. We very roughly group these developments into "generations" in order to highlight a general progression of capabilities. This is not prescriptive, and though some detailed aspects of the classification may be debatable, the basic notions hold.

1st-generation Store and Forward systems consisted of Message switching, with buffering of messages at intermediate

nodes in order to handle intermittent connectivity. There was little or no automation, intelligence, or capabilities for forming routing tables, security, network management, and handling anything but rather slow-scale dynamics. Examples include Unix-to-Unix Copy (UUCP) [3] and FidoNet [4] "In FidoNet As all modem phone numbers are published in the nodelist, point-to-point transfers are always possible. But, as store-and-forward capabilities are specified in the basic standards, email tends to be routed through a world-wide hierarchic topology and enews via a world-wide ad hoc, but generally geographically hierarchic, acyclic graph."

2nd-generation SF consist of Internet email via Simple Mail Transfer Protocol (SMTP) [5], Post Office Protocol (POP) [6], Internet Message Access Protocol (IMAP) [7], Secure/Multipurpose Internet Mail Extensions (S/MIME) [8] and so on. Key features include separation of message transfer agents, user agents, and message submission/delivery agents. There are increased capabilities for security and management. There is some (weak) separation between message format and message transfer protocols. Email servers generally operate within a well-connected environment. There are major performance problems outside this well-connected environment, because there is little diversity in message transport between nodes, and little or no improvement in dealing with dynamics.

3rd-generation SF protocols are an advance over 2nd-generation concepts. These are used to implement messaging middleware and are applied as the basis of enterprise service bus systems. There is an increased separation between message format and message transfer protocols with increased message transform / mediation capability. There has been a proliferation of proprietary formats, Application Program Interfaces (APIs) and systems, with great diversity in capabilities. Generally, there are still problems operating outside a well-connected environment, and the security mechanisms are not advanced beyond 2nd-generation ones. Examples include: JAVA Message Service (JMS) [9], Advanced Message Queuing Protocol (AMQP) [10], Streaming Text Oriented Messaging Protocol (STOMP) [11], Extensible Messaging and Presence Protocol (XMPP) [12] and Message Queuing Telemetry Transport (MQTT) [13]

4th-generation SF protocols are directed at systems with long delays and intermittent connectivity and are known as Delay/Disruption/Disconnection Tolerant Networking (DTN) [14] [15] .

There is very strong separation between format and transfer protocol as well as strong separation between format and addressing/routing. Architecturally, there are many alternatives available for transfer, addressing, and routing with heavy tailoring per each pocket of deployment. Security is possible for implementation in a limited subset of intended use cases. There are a number of experimental implementations including Interplanetary Overlay Network (ION), DTN2 and others including substantial profiles of features and capabilities [16].

5th-generation (conceptual) systems include significant amounts of longer-term storage that can be "carried" between episodes of connectivity as a main component (i.e. Store, Carry and Forward) There is an increased separation of message metadata (i.e. shipping labels) from message bodies (i.e. containers) beyond the 4th generation, enabling new approaches to security, forwarding, and possibilities for pull-based routing. Emphasis is on the "carry" function and need for strong automation in management of stored data, including support for implementing policy in Quality of Service (QoS), security, and routing. 5th-generation systems that embody the SCF concept have not yet been widely deployed. The fielded applications that could benefit from such SCF capabilities are either using point solutions adapted from prior generations of SF technology, or are lacking the strength in automation, security, and other features that the SCF technology should provide. Later sections of this document describe a generic network architecture expected to be supported by SCF / 5th-generation systems, the envisioned scenarios and use cases for these systems, more detailed comparison/contrast with existing / prior-generation systems, and a summary of lessons learned from experience with earlier systems.

III. Generic Architecture

Figure 1 illustrates a generic SCF network architecture, with the SCF agents (labeled "SCF") frequently partitioned into time-varying disconnected subsets. Depending on specifics of an individual scenario, it may be likely that some SCF agents are permanently attached to a connected network in order to provide stable gateways to/from the other SCF agents. However, in general, the system should be considered to consist of a number of primarily disconnected SCF agents at any point in time. The importance of this consideration as it relates to design, implementation, and test of potential SCF protocols is emphasized later in this document, as its effects have plagued prior SF systems.

When visualizing an SCF network, it may help to think more along the lines of a topologically dynamically-changing (mobile) relay system of agents that can periodically communicate with one another, and may be able to make at least rough predictions about their future contacts. The distribution of news and mail in the mid-1800s as described in Herman Melville's book, "Moby Dick," is a good analogy.

> "For the long absent ship, the outward-bounder, perhaps, has letters on board; at any rate, she will be sure to let her have some papers of a date a year or two later than the last one on her blurred and thumb-worn files. And in return for that courtesy, the outward-bound ship would receive the latest whaling intelligence from the cruising-ground to which she may be destined, a thing of the utmost importance to her. And in degree, all this will hold true concerning whaling vessels crossing each other's track on the cruising-ground itself, even though they are equally long absent from home. For one of them may have received a transfer of letters from some third, and now far remote vessel; and some of those letters may be for the people of the ship she now meets.....
>
> ...Every whale-ship takes out a goodly number of letters for various ships, whose delivery to the persons to whom they may be addressed, depends upon the mere chance of encountering them in the

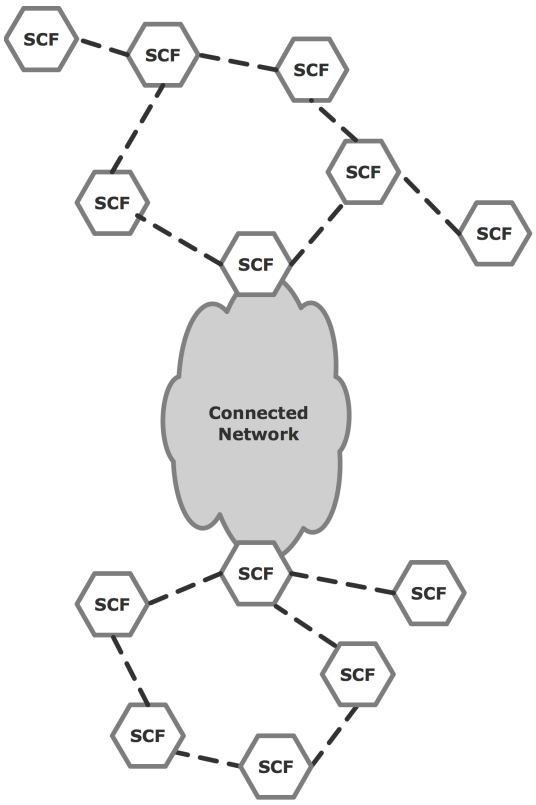

Fig. 1. Store, Carry and Forward Network

four oceans. Thus, most letters never reach their mark; and many are only received after attaining an age of two or three years or more."

Another analogy that illustrates aggregation and deaggregation are the parcel post delivery companies. Here, individual packages (containers) are delivered from a source to destinations via numerous transport mechanisms, e.g. trucks, planes, trains and boats. Along the way, these packages are aggregated into larger and larger shipping containers as they move further from the source, and then and deaggregated into smaller and smaller containers as they move closer to the destination. Such aggregation and deaggregation enable scaling of the system. There is a strong parallel between this flow of packages and the data flows seen in some of the scenarios described later in this document.

IV. Operational Considerations

Some of the key operational considerations for SCF are:

- What types of applications might be suitable to utilize SCF networking?
  - Engineering Telemetry - Accumulated over time for offline monitoring and analysis of some device or system's performance, which may be related to long-term administration of the device, but occurs in non-realtime and at a remote location. Fidelity of the received data is important, though partial delivery of data may be acceptable and more desirable than slower delivery of complete and fully accurate data. It is expected that a telemetry-sending application may operate in a fire-and-forget mode, where it does not retain data after sending.
  - Science Data Gathering - Similar to engineering telemetry, but sensor data is collected at a potentially much larger volume or over a much longer timescale. Accuracy of the delivered data is critical, and timeliness in routing may be sacrificed to provide a complete and error-free data set. Due to the size of data sets collected, having multiple copies in-flight within the network may be undesirable, and end-nodes may need to purge old data after it has been sent in order to gather new data.
  - Software Update - Numerous deployed devices that may never be able to contact an update server in realtime may need to have patches or updates deployed and activated. This can require high reliability and guarantee of eventual delivery of the data, even if the latency involved in applying the update is not extremely critical. The sender is likely to retain access to the sent update/patch perpetually, even after copies have been distributed into the network. While some acknowledgement of reception end-to-end may be desirable, this might be inferred through other means at the application level (e.g. via telemetry) rather than requiring SCF-level acknowledgement.
- In general, any distributed application where senders and receivers can operate asynchronously in non-realtime, without any real-time requirement on the infrastructure

(e.g. to do resolution of DNS names) might be able to function over an SCF service.

- What are the potential deployment environments and platform capabilities?
  - Some relevant use cases are discussed in detail in the following section. In general, the SCF agents may be either co-located or independent of the hardware/software platforms that host the end-applications. Aside from having a non-trivial amount of persistent storage, very few assumptions can be made about the SCF agent computing platforms. Typically, they will have to be embedded systems, e.g. within a device that's part of some other portable electronic system (e.g. handheld device, medical implant, avionics hardware, etc.) rather than typical workstations and servers. This means that links are expected to be (much) less capable and more time-varying than wired Ethernet, and frequent administrative access is not likely to be possible.
- What are the upper-layer user/application data-set sizes?
  - From existing systems in-use that could benefit from SCF, at least several GB of data collected onto an SCF agent between contacts with other SCF agents is possible. There are also applications where only several kilobytes of container are necessary.
- What are the traffic patterns?
  - In envisioned SCF scenarios, movement is not fully random, even for mobile ad-hoc networks, though at the very edges, it may appear random.
  - In envisioned SCF scenarios, information flow is not fully random, even for mobile ad-hoc networks.
- What type of interface between SCF agents and end applications is feasible?
  - Applications should be able to select their own globally-unique identifiers and notify SCF agents of them, along with providing proof of ownership. SCF agents may be able to notify applications of pending received data, but applications are always able to poll a SCF agent for such data as well.
- What interaction between SCF agents is expected?
  - When in contact with one another, SCF agents minimally need to be able to identify one another securely and prove that they can be trusted as relays for a given destination application. Agents should be able to indicate (or deny) forwarding of individual containers, based on exchanging their labels only.

V. Use Cases and Deployment Scenarios

There are numerous deployment scenarios for SCF systems. The following section highlights a few more common scenarios.

A. Data Mule

The Data Mule scenario illustrated in figure 2 is a common generic scenario and shows up in many deployments. In the Data Mule scenario, SCF agents communicate with each other

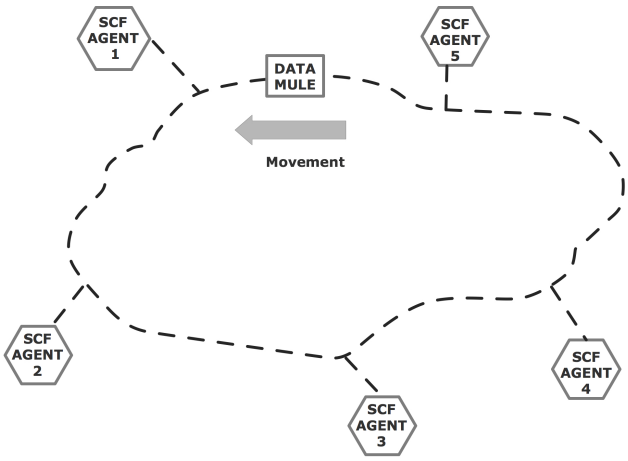

Fig. 2. Data Mule Scenario

mainly via some type of circulating entity carrying data, called the "mule". This entity may be an unmanned (or manned) aircraft, a ship, a bus, or any type of vehicle that periodically moves over the same relative area. Connectivity is likely to consist of high periods of disruption followed by short periods of connectivity over relatively high-bandwidth, low-delay, and possibly symmetric, links.

In the Data Mule scenario, connectivity is generally of the episodic variety (opportunistic). There may be one or a larger number of mules; each of which runs its own SCF agent.

Within this type of Data Mule scenario, the generic use case for SCF networking involves an application being able to push its data into containers on a SCF agent, who then interacts with the Data Mule SCF agents in order to deliver the containers to destination applications attaching to other SCF agents. In order to realize this use case, the SCF agents need to be identifiable to one another during periods of episodic connectivity, and the mule needs to somehow be able to express its expected future capability to relay containers towards the destination application.

The Data Mule is a common military scenario. It is often used to join partitioned connected networks such as groups of Mobile Ad Hoc Network (MANET). In figure 2, the SCF Agents could be concentrator points in a MANET cluster that enables communication between disjoint MANETs on the battlefield, thus enabling communications between clusters on the far ends of the communication infrastructure, the edge networks.

B. Data Gathering

The generic Data Gathering scenario is also quite common and applicable to SCF networks. Specific use cases involving sensorwebs, medical monitoring and animal tracking would fit into this scenario. Figure 3 illustrates a sensorweb where some sensors wake up and forward data through other SCF agents until they reach an SCF connected to a more powerful radio system. A gathering agent may then come by from time to time (e.g. days, weeks, months) and collect the data.

Major challenges of the use cases in a Data Gathering scenario, which go beyond those of a Data Mule, are related to

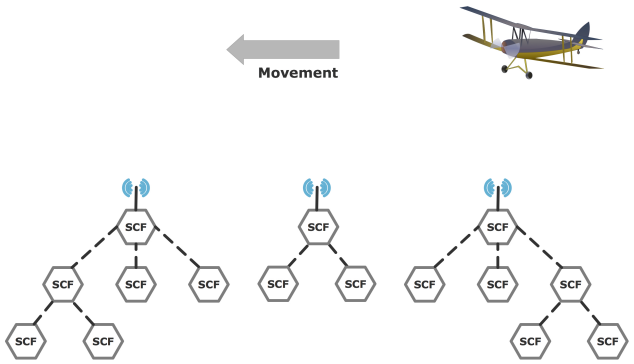

Fig. 3. Data Gathering Scenario

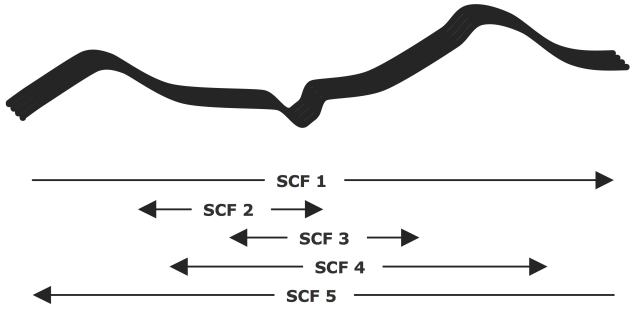

Fig. 4. Traveling the Beaten Path

the increased level of complexity in the topology between SCF agents. There is potentially less predictability, potentially more heterogeneity (or hierarchy) in SCF agent capabilities, and potentially a higher risk of routing loops or wasted resources.

The use cases for Data Gathering do, however, involve data flows that are generally either all directed from sensors up through the Gathering Agents or down from the Gathering Agents, so these still represent a sort of core network that all containers eventually go through (similar to the Data Mules).

C. Traveling the Beaten Path

There are many instances where communications may occur between entities traveling a well-worn path. In this scenario, communication is more ad-hoc than the data-mule example. The probability of encountering other SCF agents is quite high. Such scenarios include: communication in mining operations, among hikers, among boats along well traveled waterways, within the fisheries industry (the Moby Dick example) and along trade routes.

Figure 4 illustrates Traveling the Beaten Path. Consider a nomadic trade route in a third-world country. Here, SCF-1 may travel from the one end of the path to the other in one direction, while SCF-5 moves in the other direction. SCF-1 and SCF-5 will encounter all others along the way. SCFs 2, 3 and 4 only move along portions of this trade route. Most likely, none of this information is known in advance, and the movements may or may not be predictably repeatable.

D. Rapid Disruption

Many wireless networks (particularly military ones), when connected, may be relatively well-connected to a large number of other nodes in near real-time. However, individual nodes or subsets of the network may be only episodically connected because of variable link conditions given terrain, foliage, weather, jamming, or desire to evade detection. Furthermore, even when basically connected, temporary radio signal power fades can cause rapid, short, periods of disconnection. All of these lower-layer hindrances may result in short periods of disruption witnessed by applications, as well as rapid changes in network topology and the set of reachable relays.

Applications that use the Transmission Control Protocol (TCP) may perform very poorly in such environments due to TCP's congestion control algorithms - even in a non-congested environment. Furthermore, any use of the Domain Name System (DNS) is problematic in a disrupted network as one is likely to be unable to reach the DNS server. Caching DNS locally is one possible solution.

This provides some potential solutions for such networks. Routing may be capable of moving containers towards destinations via store-and-forward if the proper naming structures, addressing and routing algorithms can be developed.

Routing in a rapidly-changing topology is problematic and can result in very poor performance over wireless technologies as there is a potential to have the routing algorithms self-congest the wireless links and still not be able to converge properly.

In tactical edge military networks there may be a mix of Internet Protocol (IP) and non-IP radios. DTN Bundling, as a network overlay, provides a means to bridge IP networks and non-IP data-links together.

E. Dismounted Soldier

On the battlefield, it often occurs that a group of soldiers is on a mission and arrives via a vehicle or group of vehicles, one of which may have very good connectivity to larger networks. Once dismounted, much of the communications may be via use of that vehicle communication system as a relay or anchor. Once the group moves a significant distance from this anchor and from one other, they may become disconnected for periods of time. At this point, it becomes

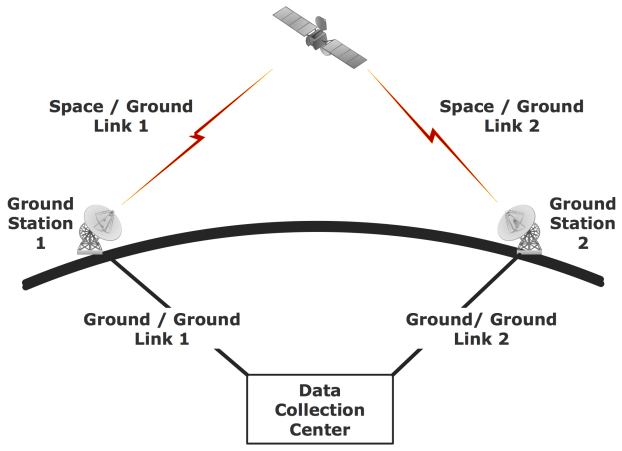

Fig. 5. Low Earth Orbit Sensor Satellite

necessary to improve communications and maintain situational awareness. This provides communication in two ways. Near-synchronous communications might be maintained via multi-hops through other agents, or in the worst case, asynchronous communications can be served sporadically when within radio contact of the anchor relay on the vehicle.

This scenario is also applicable to first responders during disaster situations where infrastructure may be severely damaged.

F. Low Earth Orbiting Sensor Satellite

The Low-Earth Orbit (LEO) scenario shown in figure 5 is for a sensor satellite communicating directly with ground terminals. One such network is described in reference [17]. Note, in this scenario, no geostationary relay satellite is involved. Here, the contact times may be known in order to direct the satellite transmitter to turn on. Some type of automated hailing could also be used.

LEO is a low-propagation-delay environment of less than ten milliseconds delay to ground, with long periods of disconnection between passes over ground stations. Contact times consist of a few minutes per ground station for Earth satellites orbiting at a couple hundred kilometers altitude (100 minutes per orbit). Thus, the more ground systems that are available, the greater the potential for contact. The ground stations are connected across the terrestrial Internet, or other backbones.

In this scenario, the agent onboard the satellite does not need to perform forwarding of received containers. The satellite is a source for sensor data and may be a sink for command data.

The main reason to use SCF in this scenario is to provide a standardized relaying technique and to decouple the control loops between the space/ground link and the ground/ground link.

There are numerous companies and systems today that transfer extremely large sensor data sets from LEO to ground without a standardized method. Those data sets are in the multi-gigabyte region and growing. However, they are using protocols and implementations that are not compatible with one another, and which require them to go through various levels of customization per-use.

VI. Consideration of Existing Technologies

In this document, we characterized DTN-based systems as 4th-generation SF rather than systems. Several aspects of DTN are highly desirable and applicable to SCF. DTN utilizes "bundle agents" that are similar to the "agent". Several DTN routing protocols, that exist at varying levels of maturity, can work well for individual scenarios that have been outlined. For instance, Contact Graph Routing is particularly useful in scenarios where future connectivity is predictable/known ahead of time. The container aggregation and deaggregation bears some surface similarity to bundle fragmentation operations in DTN, but there are major differences. Containers are intended to be aggregatable within the network, even if they are not portions of the same original container from the application. Additionally, some applications (e.g. science data collection) may find (optional) partial reception of subsets of large containers that have been deaggregated into smaller containers, to still be useful, whereas DTN only delivers entire (reassembled) bundles. This does require the data formats used by such applications to be self-synchronizing, so that they can be parsed, but this is an issue for the application. scenarios require some features that are not yet a part of the DTN specifications:

- The ability to avoid Denial of Service (DoS) by propagating an application's permit/deny filters to agents.
- The ability to generate and prove ownership of globally-unique application identifiers.

DTN is the targeting of operation over very high-latency data links. SCF does not explicitly attempt to operate over such links, though it may end up being possible. Since these links are mostly only applicable to deep-space scenarios with small numbers of nodes, motivations do not include high-delay.

This document also identified JMS and Message Queuing Telemetry Transport (MQTT) message-broker systems as 3rd-generation SF rather than systems. JMS "messages" transferred between "brokers" and applications are similar to the containers transferred between agents and applications. JMS offers both point-to-point (unicast) and publish-subscribe (multicast) models of communication. JMS uses named "queues" (in the point-to-point model) or "topics" (in the publish-subscribe model) in order to identify destinations. JMS brokers often implement a "durable" messaging service that allows messages (and queues) to persist even when the applications that created them (or need to receive them) are disconnected from the broker. scenarios require some features that are not yet really reflected within the JMS specifications:

- Multi-hop relaying among brokers and secure propagation of information about the queues/topics present or acceptable is not standardized.
- Communication of an application's desired permit/deny filters on queues that it owns is not standardized.

JMS is an API and not a protocol standard. This is the primary hurdle in using JMS to support , as the wire-protocols and other mechanisms used in a particular JMS implementation are not necessarily compatible with others. requires full vendor/platform interoperability in order to be cost-effective to pursue in preference to point-solutions. JMS

is also significantly focused on transfer of Java objects rather than generic containers of bytes as should be.

One of the biggest challenges to using existing systems (whether they be DTN implementations, JMS products, or some others) is that most have been designed to include a multitude of additional (optional) features and this results in, at best, limited compatibility between implementations. For instance, the DTN Bundle Protocol is an excellent platform for experimentation due to its flexibility and ease of defining new "blocks" to implement different functionalities. DTN has been used or demonstrated in a wide range of scenarios with differing needs, including simulated military exercises, connecting people in remote regions, moving data from LEO spacecraft, deep-space missions, mining operations, and others. However, individual implementations have been developed to support distinct subsets of the defined blocks, identifier schemes, and algorithms that suit the unique properties of their pet environments. Developing, and then maintaining, a baseline for compatibility has not been a primary concern. For an operational system, a baseline profile of the required functionality would need to exist, which could be present across the spectrum of vendors. For DTN, this type of profile is not present in the existing systems to a level that would enable the scenarios described in this document. Saving energy and running on very small devices (e.g. sensors and embedded medical devices) also motivates having such a profile that could aid in developing very small, yet fully compatible, implementations.

Message Queuing Telemetry Transport (MQTT) is a lightweight protocol used for publish/subscribe messaging between devices. MQTT was designed for low-bandwidth, high latency networks while attempting to ensure reliability of delivery. A major design criteria was simplicity - must be simple to implement and must not add too many "bells and whistles" that would complicate implementation, while still providing a solid building block which can easily be integrated into other solutions. MQTT is designed to handle frequent short periods of network disruption using a technique called "Last Will and Testament". Although not part of the specification, MQTT has been modified to operate in multi-hop environments.

VII. Characteristics of Information

Since information has to be transported and stored, in an use case, it is important to be aware of the key characteristics of the information being acted upon. All information has a source and one or more eventual destinations. All application information ready to be sent, has a size that may be very small or quite large (several bytes to multiple gigabytes). Size is important because storage is not unlimited in either the source application's system nor in the relays, and because transmission bandwidth and contact times limit the amount of data that can be sent during any given contact time.

Information may have security restrictions placed on it - sensitive or restricted (for your eyes only), and in some cases this may be handled at the application layer, as is done by securing email.

All information has a useful lifetime. It may be very short (seconds) or very long (days, weeks, years). Regardless, it is only the users of the information that know what the real useful lifetime is, and it is the application that would be required to set that lifetime. With the exception of specific cases, it is not at all clear that the application can generally make that decision.

Often data has a "freshness" characteristic. For a given application, data that is more recent (fresher) is often of greater value that data that is older (stale). In such cases, it may be more important to forward the most recent data, rather than the data that is near the end of its useful lifetime. One might even purge the system of stale data. One trivial example that illustrates why data freshness is important would be reception of stock quotes. Obviously, one would not expect such systems to be used for commodities trading due to disruption and ordering issues (assured timeliness). Rather, applications such as sensor data transmissions, software updates or distributed security-key databases are more amenable to deployments.

VIII. Network Management

Network management is needed to keep the network running smoothly. It is required for system configuration and maintenance, and monitors the system to determine faults, performance, security issues and accounting. From the scenarios presented, it appears that network management is likely to be per-scenario, and may be effectively accomplished out-of-band. For example: in the Data Mule scenario, one may manage the data mule, but not the edge systems. In the Data Gathering scenario, one is likely to preconfigure the remote sensor nodes and only manage the data-gathering and perhaps the data concentrators, the ones with high-power radios.

This does not imply the network management could not or should not be performed in-band; only that it may not be required.

Since resources (e.g. bandwidth, transmit power, and storage) are a precious commodity in networks, policy that manages those resources is expected to be a major component of system configuration. For example: a particular agent may restrict particular information sources to limited storage space and limited storage time. Such policy may restrict all information to a limited storage time in order to purge stale containers. Also, particular sources may get preferential treatment per peering agreements.

IX. Lessons Learned Summary

There are numerous lessons to be learned from previous deployments of MANETs and 4th-generation store and forward networks such as DTNs. Some of the more critical and important pieces of knowledge are listed below:

- systems are generally connected via radio networks. Some radio systems may take far less power to listen than to transmit, though this varies by individual link technology. Wasted transmission is wasted power on a wireless system and can quickly drain a battery. The problem is compounded for devices whose entire lifetime is determined by their battery (e.g. non-rechargeable

sensor nodes). Thus, reducing wasted transmissions is high desirable.
  - The ability to reactively fragment large data sets en-route is highly desirable. This has been demonstrated in DTN experiments.
  - Routing loops will not be caught by layers below. It is imperative that data dies naturally and quickly so as to not waste bandwidth or transmission power. Such loops have been encountered in early experiments with DTN overlays, and are correctable.
  - It is highly desirable for the sender to know early in a transmission whether or not the receiver will accept the data. This permits a savings in power and optimization of network capacity usage. For instance, in DTN experiments with large bundles, the entire large bundle may be sent, only to be discarded due to security, resource scarcity, or other issues.
- Disconnected networks are difficult, if not impossible, to globally synchronize state across.
- When investigating the use of DTN bundle lifetimes in DTN deployments and implementations, we have found that the lifetimes have generally been set to match the duration of the experiment. There are instances where some finer granularity has been deployed such and in the Defense Advanced Research Project Agency (DARPA) Wireless Network after Next (WNAN) where a small number of choices in lifetimes of minutes or hours are used depending on the nature of the data. The DTN bundle lifetime can be used for two purposes: expedited forwarding for end-applications and purging stale information from the relays. Thus, the real requirement should be the ability to expedite forwarding of priority containers and purge stale containers from the system, but not necessarily to specify time-sensitivity on a per-second basis. There may be other means to accommodate this requirement without having to burden the agents with the management and synchronization of notions of "time" - particularly per container - which has been burdensome in DTN use.
- Managing fine-grained notions of time in a protocol is difficult and adds considerable complexity.
- Having a relay protocol be time-dependent opens it up to security vulnerabilities.
- Having a relay protocol be time-dependent complicates use of that protocol to synchronize the system - even for coarse synchronization.
- It is highly desirable for a receiving agent to determine early within a transfer whether or not to accept the data. Large data sets utilize significant processing and storage resources for data that may end up being discarded due to security, resource constraints, or other policy issues.
- It is highly desirable to keep forwarding tables small, and make forwarding decisions ahead of time for predicted contacts. Book-keeping type of processing while forwarding a large number of small containers can overload the processing system.
- Testing should be thorough and include exercising both the storage and forwarding systems. Failure to do so will lead to erroneous results . Thus, any testing and validation should exercise both the storage and forwarding mechanisms of the implementation. To do otherwise may lead to misleading results.

X. Security Considerations

Applications need to authenticate to an agent before they can send or receive containers.

Authentication of agents to one another needs to be tackled before advertisements of forwarding capability can be acted upon.

Bandwidth, Storage, and Processing Power are precious resources in an SCF network. In order to reduce DoS vulnerabilities and properly allocate resources, an SCF agent should be able to determine whether or not to act on a container based solely on the Shipping Label.

Applications should be able to limit DoS by expressing explicit desires to a serving agent for/against certain traffic selectors. It may be beneficial for this information to propagate between agents, though it should be recognized that any dynamics in these preferences causes a risk of data loss due to lack of synchronization of the filter rules.

While some aspects of Public Key Infrastructure (PKI) may be applicable, PKI itself is not because, in general, PKI requires connectivity. Public-Keys with caching may be applicable; however, this would require at least some coarse network synchronization.

XI. Requirements and Expectations

The following subsections describe the requirements for a Store, Carry and Forward (SCF) protocol, and the expectations placed upon the SCF agents and SCF applications.

A. Design Considerations

The following design considerations are explicitly stated with a goal of keeping the protocol simple. (Anyone can make things more complicated!)

- Do not overload the relaying protocol. Keep It Simple.
  - Keep network management functions separate from the relaying protocol.
  - Content Based Networking is different than SCF.
  - SCF can be used to move content, but should not be considered an in-network content store.
  - Rationale: Separation allows for independent development and optimizations.
- The SCF protocol MUST NOT rely on time synchronization between applications or relaying agents.
  - It is very difficult, if not impossible, to synchronize disconnected networks. Furthermore, if the protocol requires synchronization to work, it can never be used to synchronize a system - even for coarse synchronization. In addition, reliance on absolute time creates security vulnerabilities.

- Protocol options make interoperability hard. Options are often used as a placeholder for fixing a bad design.
- Naming and addressing are key to security and scalability.
- Addressing should be topological (location dependent). This enables aggregation of the routing locator space and improves scalability for the routing system.
- Strive to limit the size of the forwarding table. Large forwarding tables place a great burden on the SCF processing system. There is always a limit for any Central Processing Unit (CPU). The further one is removed from reaching that limit, the better.

B. Protocol Requirements

The following are a list of requirements for a SCF Protocol. The requirements are specifically written in general terms. The intent is to identify what is required, not how to solve the requirement. The requirements are in no particular order of precedence, but are numbered in order to aid in referencing for discussion.

SCF Agent Requirements and Agent operation expectations have been intentionally separated, as the SCF Agent requirements are more policy-based than protocol-based. However, one needs to understand both in order to effectively implement the SCF protocol.

proto 1: The SCF relaying protocol MUST be able to handle data sets that are very small (several bytes) and very large (several gigabytes).
- Rationale: SCF is useful for very small, simple, low-power, low-processing minimally-capable sensor systems, as well as for more capable high-end data mules. In a simple sensor-web one may be moving extremely small containers of information on the order of bytes; whereas later onward delivery by a data mule may be moving containers containing gigabytes of data.

proto 2: The SCF protocol MUST permit SCF agents to be able to aggregate containers.
- Rationale: Aggregation will reduce forwarding table size and enable pre-processing of forwarding queues. Without aggregation, the SCF agent processing capabilities can be quickly overwhelmed - particularly for a large number of small containers - even if those containers are destined for the same location. Aggregation and Deaggregation enable efficient shipping of information through a SCF network from a variety sources to a common destination by continually recombining containers as the information moves through the relay network.

proto 3: The SCF protocol MUST permit SCF agents to be able to deaggregate containers.
- Rationale: Deaggregation allows subcontainers to be removed from larger aggregated containers and either shipped separately due to contact limitations, or spread out to multiple other relaying SCF agents in parallel.

proto 4: The SCF protocol MUST permit SCF agents to be able to reactively fragment a container.
- Rationale: It is often not possible to determine how long a contact time will be between SCF agents. In such instances, which may be the norm, one cannot determine the transport capacity and may only be able to transfer a portion of a container before the contact ends. In order to improve transport efficiency and effectively utilize the radio link, one should not have to retransmit what has already been received.

proto 5: The SCF protocol SHOULD permit SCF agents to be able to proactively fragment a container.
- Rationale: It may be possible to a priori know the transport capacity between SCF agents. In such instances, one may determine that an entire container could only be transfer between agents if it is divided into smaller units. In other cases, a SCF agent may wish to limit the size of containers as a matter of policy. In either of these cases, proactive fragmentation would be useful. However, it would be more desirable for the application to limit the size of the container if at all possible, rather than having this done by the SCF agent.

proto 6: An SCF protocol MUST implement reliability on the Shipping Label, and a damaged Shipping Label MUST NOT propagate to further SCF agents or have its container further propagated or delivered to applications.
- Rationale: An SCF agent needs to be able to determine if the shipping label is damaged in order to prevent misdelivery of data, waste of resources (storage, battery, network capacity, etc.), and other suboptimal results of operating on flawed forwarding information.

proto 7: An SCF protocol MUST be capable of implementing reliability on the Shipping Container.
- Rationale: An SCF agent must be able to determine if the shipping container is damaged. A damaged Shipping Container MAY be discarded along with the associated Shipping Label. Note, user of reliability on the Shipping Container is not mandatory, but the ability to have such capability is.

proto 8: The SCF protocol security implementation MUST authenticate the Shipping Label independent of the Shipping Container.
- Rationale: The ability to authenticate data sources and control resource usage early on is critical to reducing vulnerabilities to denial-of-service attacks, whether intentional or unintentional. For large containers, if the entire container had to be received and processed before a determination could be made as the the source of the Container, multiple resources would be wasted including bandwidth, processing cycles and storage.

proto 9: The SCF protocol security implementation MUST work with reactive fragmentation.
- Rationale: For medium- and high-end SCF sys-

tems, the ability to authenticate data sources and control resource usage is critical. Likewise, reactive fragmentation may be quite common and has been shown to be invaluable in transporting large data sets [18].

proto 10: The SCF protocol security implementation MUST have a security policy database to control resources.

- Rationale: Once a data source is authenticated, the security policy will determine what type and amount of resources that source can use, as well as possibly the forwarding priorities. It is anticipated that SCF systems will have different peering arrangements with different entities (e.g. peers, groups, or organizations).

proto 11: The SCF protocol MUST be able to separate the Shipping Label from the Shipping Container

- Rationale: The SCF agent must be able to determine whether or not it wishes to receive or store a container prior to receiving the entire container. This reduces denial-of-service vulnerabilities and enables efficient use of radio and system resources.

proto 12: The SCF protocol MUST have a mechanism that enables data to die naturally.

- Rationale: Data should die naturally to avoid routing loops at the SCF layer. Routing loops at the SCF layer cannot be eliminated by lower-layer mechanisms (i.e. and Internet Protocol version 6 (IPv6) hop count will not correct an SCF routing loop).

proto 13: The SCF protocol MUST have a naming mechanism that specifies the application and instance to which the content is bound.

- Rationale: This naming mechanism is necessary in order for the SCF agent end system to pass the Shipping Container content to the proper instance of a given application since multiple instances may be invoked at any given time.

proto 14: The SCF protocol MUST have a Quality-of-Service (QOS) mechanism to expedite forwarding and to handle storage lifetimes.

- Rationale: Past experiences with other store-and-forward technologies such as DTN [15] have shown that it is very difficult for many applications to determine how long the useful life of data is. Rather, bundle lifetimes have been set either arbitrarily or rather coarsely (e.g. short, medium, forever) - see ¡eref target="http://www.dtnrg.org/wiki/DtnBone"¿Bundle Lifetimes¡/eref¿. QOS will enable and SCF to expedite, store and purge data on a much more coarse scale than the use of absolute or relative time. Such QOS policies could be a configuration setting within individual SCF agents, or within an SCF network. This greatly simplifies the protocol processing as well as aggregation and deaggregation of containers.

proto 15: The SCF protocol MUST have a mechanism for a receiving system to acknowledge reception of a container from the sending system (i.e. hop-by-hop acknowledgements).

- Rationale: This allows a sending system to release the container if it so desires, thereby improving resource usage.

proto 16: The SCF protocol MUST have a mechanism to notify a sender that the container will not be processed.

- Rationale: If the agent's policy states "Do not accept" for any possible reason, it is important to inform the sender as soon as possible (ASAP) that the container will not be accepted, to allow the sender to stop transmission and determine a different route for that container. Note, there may be security reasons not to provide this information, but in generally such a response SHOULD be sent.

proto 17: The SCF protocol SHOULD have a mechanism that enables one to identify fresh versus stale content for a given flow.

- Rationale: Fresh data is often of far greater value than stale data. The ability to identify fresh data and either replace the stale data with fresh, or send the fresh data first, is highly desirable in order to optimize resource usage - particularly storage and bandwidth.
- Comment: There appears to be a desire in many instances to proactively create fixed bundle sizes in DTN and then what the application to put them back in order. With proactive fragmentation, this is possible and there is a mechanism to allow reordering. With straight bundling, this is problematic as there is no such formalized standard sequencing (i.e. sequence numbers).

C. Agent Requirements and Expectations

The following are a list of requirements for an SCF Agent. The requirements are in no particular order of precedence, but are numbered in order to aid in referencing for discussion.

agent 1: An SCF agent MUST NOT be required to implement SCF security. Security must be optional.

- Rationale: Simple devices such as sensors may wish to utilize the SCF protocol, but have neither the need for security, nor the processing capability to implement SCF security.

agent 2: An SCF agent MAY implement reliability on the Shipping Container.

- Rationale: An application may or may not care if the contents of the container arrive without modification. For example, protecting a large image file from a bit flip may not be considered as important as reducing the processing overheard of creating and checking reliability on the Shipping Container.

agent 3: An SCF agent MUST hold onto a container until it can either be transferred or QOS policy indicates its useful lifetime has expired or storage resources reach

a level that requires some purging of containers based on policy.

- Rationale: The sender expects the receiver to do its best to forward the container, and MAY release the container upon notification from the receiver that the container has been received. If the receiver does not plan to hold onto the container, it SHOULD send a notification to the sender stating such.

agent 4: An SCF agent MAY remove a container once it receives notification from the next hop SCF that the container has been delivered.

- Rationale: The ability to release containers enables efficient use of storage resources. Note, some deployments and some routing protocols MAY mandate that the agent retain a container even after a successful transfer. In such deployments, containers would likely be removed based on a retention policy which may be based on QOS.

agent 5: An SCF agent SHOULD NOT accept a container if it has no intention of giving a best effort to forward the container.

- Rationale: The sending SCF's default expectation is that, if accepted, the receiving SCF will do its best to forward the container. This allows the sending SCF, if so desired, to purge the container from its storage with some confidence that the container will be delivered.

agent 6: An SCF agent SHOULD implement a policy system that controls resources. Such a policy system MAY include the filters described below.

- Rationale: Resources including bandwidth and storage storage are precious commodities that need to be controlled. Various SCF deployments are expected to have vastly different capabilities and needs. For example, an SCF science sensorweb may have not need for security, while implementing a policy that basically says "Accept Everything", because all containers are know a priori to be small and the deployment is a closed network. Other deployments may consist of high-end SCF agents supporting multiple organizations and transferring and storing Gigabytes or more of information. The ability to tune the policies to fit the deployment makes such deployments realizable.

a) What volume (size) container will be accepted.
   - Rationale: Storage resources are not infinite. It is likely policy will limit container size and/or overall memory allocations per source, address range, or other filters. In addition, some SCF agents may have limited processing and not be willing or capable of handling extremely large containers

b) What sources are permitted to use resources and how much resource?
   - Rationale: Resources including bandwidth and storage are precious. It is anticipated that peering arrangements will exist to populate this database. Not every source may be permitted to utilize the resources.

c) What destinations are permitted to use resources and how much resource?
   - Rationale: Resources including bandwidth and storage are precious. It is anticipated that peering arrangements will exist to populate this database. Not every destination may be permitted to utilize the resources.

d) Prioritized container delivery.
   - Rationale: It is anticipated that peering arrangements will exist to populate this database. Some peers are likely to be given preferential treatment, while others may be serviced only after all commitments have been met, regardless of QoS (e.g. the general's containers are processed before the private's; the organization who owns the SCF agent gets preferential treatment over all other organizations.

e) A mapping of QoS to retention lifetime and forwarding priority.
   - Rationale: A coarse-grained retention policy is anticipated. Such granularity may be minutes, hours, days, forever (until resources become scarce and memory must be released). This alleviates the need for actual lifetime settings within the SCF protocol and allows various deployments to be uniquely configured.

agent 7: If security is implemented, when coming in contact with one another, adjacent SCF agents MUST minimally be able to identify one another securely and prove that they can be trusted as relays for a given destination application.

- Rationale: Such a mechanism is necessary to prevent hijacking of information. Also, aggregation and deaggregation may be implemented along a container's route. Trust between forwarding agents must be established to enable this.

agent 8: SCF Agents MUST be able to indicate (or deny) forwarding of individual containers, based on exchanging their shipping labels only.

- Rationale: This allows for efficient use of RF resources as well as reducing DOS vulnerabilities. It the SCF Agent had to process an entire Container prior to denying acceptance, a malicious entity could easily perform a DOS attack by sending extremely large containers which would have to be stored and processed by the receiving SCF prior to rejection.

agent 9: SCF Agents MAY notify applications of pending received data.

- Rationale: If the SCF agent knows it is bound to an application and can notify the application of pending received data, this could improve the application's operations.

D. Application Requirements and Expectations

The following are a list of requirements for Applications that would be used over a SCF network. The requirements are in no particular order of precedence, but are numbered in order to aid in referencing for discussion.

appl 1: Applications SHOULD be designed to operate in a disconnected systems.
- Rationale: Applications that have been designed assuming a connected network are likely to break.
- Rationale: Streaming may work, but should not be encouraged as streaming applications with reasonably significant volumes of traffic are likely to only work for connected systems or very short fades. Such systems probably do not need SCF.

appl 2: Applications MUST be able to select their own globally-unique identifiers and notify SCF agents of them, along with providing proof of ownership.
- Rational: A given service may run at one or more nodes, and may need to move from one node to another without losing its identity as a service.
- A given node may be connected to one or more network attachment points, and may need to move from one attachment point to another without losing its identity as a node.
- A given pair of attachment points may be connected by one or more paths, and those paths may need to change with time without affecting the identity of the attachment points [19].

appl 3: Applications MUST be able to poll a SCF agent for pending received data.
- Rationale: Applications are the only ones that can keep track of the shared state between sender and receiver. The Application cannot expect lower layers such as the SCF agent to fully understand its needs.
- Rationale: This eliminates putting undue burden on the SCF, and ensures interoperability to specify a known operational expectation.

XII. Testing Store, Carry and Forward Systems and Protocols

In RFC760 [20], one can find what has become know as Postel's Law or the Robustness Principle, "In general, an implementation should be conservative in its sending behavior, and liberal in its receiving behavior." This rule was originally targeting protocol implementation. A corresponding rule for testing may be, "If you claim the protocol can do it, you have to prove it - test it."

Conversely, being able to simply ping an end system does not indicate the network is fully functional. It just means that there is connectivity and the potential for the network to be fully functional.

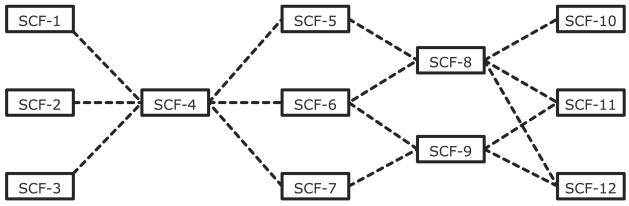

Fig. 6. SCF Test Network

The intent of this section is to establish thorough, repeatable, tests that will fully exercise a SCF system. Past experience has shown that testing of SCF systems is too often inadequate. For example, tests have been performed on SCF systems in fully-connected, high-bandwidth networks where only forwarding would be exercised, or the traffic would be so minimal as to never tax the storage or queueing. Such tests are valid as a starting point, but insufficient to determine that a protocol or implementation will working properly in a reasonably-scaled deployment.

A secondary motivation is to improve implementations by providing a known test environment. Knowing some possible ways that the protocol and system will be evaluated may help establish how the code is developed, as well as identifying hooks for monitoring particular processes.

A. Test System

Figure 6 illustrates a generic testbed for testing may aspects of the SCF protocol. The systems consists of 12 SCF agents and 16 links. Any or all of the links may be disconnected at any given time. Even though the system is simple, some complexity is necessary because the system must accommodate testing of aggregation, deaggregation, and fragmentation with multiple container flows of various sizes and priorities.

B. Test Requirements

The following are a list of SCF protocol test requirements. This lists is likely incomplete, but at least provides a starting point. The requirements are in no particular order of precedence, but are numbered in order to aid in referencing for discussion.

Test 1: While the SCF Protocol SHOULD be capable of handling all environments, the implementation may not However, the implementation SHOULD be tested over the environment it is meant to operate.
- Rationale: Delays and disconnection times may be specific to a given deployment and the implementation may be design for that particular deployment environment.
- Rationale: Container sizes may be quite limited in some deployments such as a sensor web or may be quite large in other instances such as a remote imaging sensor satellite.

Test 2: The implementation should be tested with traffic density and traffic patterns that are anticipated in the particular deployment

- Rationale: Communications is rarely if ever random. Testing with random data patterns can be very misleading. For example, in a warfighter deployment, there may be some background random communications between peers. While the majority of traffic may be from the edge to a concentration point such a a command center.

Test 3: Although SCF routing protocols are not specified in this document, if a SCF network deployment has aspects of mobility, that mobility MUST be emulated or implemented in the test network.

- Rationale: Failure to test a routing protocol designed to handle a time-varying SCF topology will likely result in misleading results.

Test 4: For all implementations - expert perhaps the simplest of sensor webs - fragmentation, aggregation and deaggregation SHOULD be tested.

- Rationale: SCF environments, buy there very nature, are unpredictable (with the perhaps the exception of very simple space-based networks). As such, one should not assume that container flows will not experience situations where fragmentation, aggregation and deaggregation will result.

Test 5:
- Rationale:
- Rationale:

XIII. Conclusion

The conclusion goes here.

Acknowledgements

Much work builds on lessons learned from the work performed by the Internet Research Task Force (IRTF) DTN Research Group.

Work on this document at NASA's Glenn Research Center was funded by the NASA Glenn Research Center Innovation Funds.

References

[1] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119 (Best Current Practice), Internet Engineering Task Force, Mar. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2119.txt

[2] J. Day, *Patterns in network architecture: a return to fundamentals*. Prentice Hall, 2007.

[3] (December, 2013) Unix-to-unix copy. [Online]. Available: http://en.wikipedia.org/wiki/UUCP

[4] (December, 2013) Fidonet. [Online]. Available: http://www.fidonet.org/

[5] (December, 2013) Simple mail transfer protocol. [Online]. Available: http://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

[6] (December, 2013) Post office protocol. [Online]. Available: http://en.wikipedia.org/wiki/Post_Office_Protocol

[7] (December, 2013) Internet message access protocol. [Online]. Available: http://en.wikipedia.org/wiki/Internet_Message_Access_Protocol

[8] (December, 2013) Secure/multipurpose internet mail extensions. [Online]. Available: http://en.wikipedia.org/wiki/S/MIME

[9] (December, 2013) Jms/openwire. [Online]. Available: http://en.wikipedia.org/wiki/Java_Message_Service

[10] (December, 2013) Advanced message queuing protocol. [Online]. Available: http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol

[11] (December, 2013) Streaming text oriented messaging protocol. [Online]. Available: http://en.wikipedia.org/wiki/Streaming_Text_Oriented_Messaging_Protocol

[12] (December, 2013) Extensible messaging and presence protocol. [Online]. Available: http://xmpp.org

[13] "Message queuing telemetry transport," 2013 December. [Online]. Available: http://mqtt.org/wiki/

[14] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," RFC 4838 (Informational), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4838.txt

[15] K. Scott and S. Burleigh, "Bundle Protocol Specification," RFC 5050 (Experimental), Internet Engineering Task Force, Nov. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5050.txt

[16] (2013, December) Dtn research group. [Online]. Available: http://www.dtnrg.org/wiki/

[17] L. Wood, W. Ivancic, W. Eddy, D. Stewart, J. Northam, C. Jackson, and A. da Silva Curiel, "Use of the delay-tolerant networking bundle protocol from space," in *Proceedings of the 59th Astronautical Congress, Glasgow. IAC*, 2008.

[18] W. Ivancic, P. Paulsen, D. Stewart, W. Eddy, J. McKim, J. Taylor, S. Lynch, J. Heberle, J. Northam, C. Jackson *et al.*, "Large file transfers from space using multiple ground terminals and delay-tolerant networking," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010, pp. 1–6.

[19] J. Saltzer, "On the Naming and Binding of Network Destinations," RFC 1498 (Informational), Internet Engineering Task Force, Aug. 1993. [Online]. Available: http://www.ietf.org/rfc/rfc1498.txt

[20] J. Postel, "DoD standard Internet Protocol," RFC 760, Internet Engineering Task Force, Jan. 1980, obsoleted by RFC 791, updated by RFC 777. [Online]. Available: http://www.ietf.org/rfc/rfc760.txt

William Ivancic Mr.Ivancic has over thirty years experience in network and system engineering for communication applications, communication networking research, state-of-the-art digital, analog and RF hardware design and testing. He currently is a senior research engineer at NASA?s Glenn Research Center. His work areas include network centric technologies for space, aeronautics and terrestrial systems. He has lead research efforts to deploy commercial-off-the-shelf (COTS) technology into NASA missions. Of particular interest is large scale, secure deployment of mobile networks including mobile-ip and mobile router technology. Mr. Ivancic?s recent areas of research include: high-speed reliable data transport protocols, store-carry-and-forward protocols, and adaptive dynamic networking including cognitive networking. He has recently become involve with developing the detailed communication architecture for advanced extravehicular mobile units (a.k.a. spacesuits).

Mr. Ivancic is also principle of Syzygy Engineering, a small consulting company specializing in communications systems and networking as well as advanced technology risk assessment. Mr. Ivancic is currently performing research and development on Identity-based security and key and policy management and distribution for tactical networks - particularly mobile networks.

Wes Eddy Wesley Eddy is Chief Technologist at MTI Systems, where he supports contracts with NASA and other agencies. He has an MSc degree in Computer Science from Ohio University, where he studied modifications to TCP congestion control. For NASA, he has worked on a number of space and aeronautics research projects, including the first demonstration DTN in space onboard the UK-DMC satellite and several studies involving space-based networking and protocol enhancements needed for the space environment. He has also worked as a systems engineer on NASA teams including the SCaN-Constellation Integration Project (SCIP), the Space Network (SN) Ground Segment Sustainment (SGSS) project, the Altair lunar lander, and the SCaN Testbed. He is active in the Internet Engineering Task Force (IETF), where he co-chairs the Active Queue Management and Packet Scheduling (AQM) working group. In the past, he was an IETF Area Director and also co-chaired the IETF TCPM working group and the Internet Congestion Control Research Group (ICCRG).

Alan Hylton Alan Hylton is a communications researcher at NASA GRC specializing in Delay Tolerant Networking (DTN) and deep-space optical communications. He is the technical lead for DTN at GRC, and is the networking lead and test-bed lead on the Integrated Radio Optical Communications (iROC) project. At NASA, his beginnings were in the biosciences and technology group. His background is in pure mathematics.

Dennis Iannicca Mr. Iannicca is a computer engineer at NASA Glenn Research Center. He is currently researching delay and disruption tolerant networking technology to integrate into terrestrial and space communications systems. As part of NASA's Integrated Systems Research Program, he is also conducting communication security research to provide recommendations to the FAA and industry on how to securely integrate civil Unmanned Aircraft Systems (UAS) in the National Airspace System (NAS).

Joseph Ishac Joseph Ishac is a computer engineer for NASA Glenn Research Center in Cleveland, Ohio. He has earned both his bachelor's and master's degrees in computer engineering from Case Western Reserve University, where he studied the impact of transparent network proxies. He is actively involved the advancement and development of networking protocols, using both simulation environments and experimental testbeds. His current work areas include advancing satellite data link technology, including use of Low Density Parity Check Codes (LDPC). He is also working on the integration of Unmanned Aerial Systems (UAS) into commercial airspace. Joseph research interests lie in the field of computer networks and architectures, IPv6, security, and transport protocols.

List of Acronyms

AMQP - Advanced Message Queuing Protocol
API - Application Program Interface
CPU - Central Processing Unit
DARPA - Defense Advanced Research Project Agency
DNS - Domain Name System
DoS - Denial of Service
DTN - Delay/Disruption/Disconnection Tolerant Networking
IMAP - Internet Message Access Protocol
ION - Interplanetary Overlay Network
IP - Internet Protocol
IPv6 - Internet Protocol version 6
IRTF - Internet Research Task Force
JMS - JAVA Message Service
LEO - Low-Earth Orbit
MANET - Mobile Ad Hoc Network
MQTT - Message Queuing Telemetry Transport
PKI - Public Key Infrastructure
POP - Post Office Protocol
QoS - Quality of Service
RFC - Request For Comment
SCF - Store, Carry and Forward
SF - Store and Forward
S/MIME - Secure/Multipurpose Internet Mail Extensions
SMTP - Simple Mail Transfer Protocol
STOMP - Streaming Text Oriented Messaging Protocol
TCP - Transmission Control Protocol
UUCP - Unix-to-Unix Copy
WNAN - Wireless Network after Next
XMPP - Extensible Messaging and Presence Protocol